# TP2 : Optimisation des Scripts Python avec Turtle 🐢✨

**Bienvenue dans l'aventure Turtle 2.0 !** 🚀

Maintenant qu'on a dessiné des formes basiques, on va transformer ton code en machine de guerre : plus court, plus flexible, et plus fun !
On va utiliser des boucles `for` pour éviter les répétitions (adieu, copier-coller !), des `input()` pour laisser l'utilisateur décider (comme un boss), et des fonctions `def` pour réutiliser ton code comme un pro.
Prêt à level up ? Allons-y ! 🎉

## 1. Optimisation avec Boucle for 🔄

Imagine : ton code répète les mêmes instructions ? C'est comme manger la même pizza tous les jours... ennuyeux ! Une boucle for gère ça en un clin d'œil. Voici un exemple avec le rectangle – compare les deux versions et teste-les pour voir la magie opérer !
**Sans boucle `for` (le vieux style, long et répétitif) :**

| **Sans boucle for : (le vieux style, long et répétitif)** | **Avec boucle for (plus court, plus élégant – teste-moi ça !) :** |
|---|---|
| `from turtle import *`<br><br>`forward(100)`<br>`left(90)`<br>`forward(50)`<br>`left(90)`<br>`forward(100)`<br>`left(90)`<br>`forward(50)` | `from turtle import *`<br><br>`for i in range(0, 2):`<br>`    forward(100)`<br>`    left(90)`<br>`    forward(50)`<br>`    left(90)`<br>indentation<br>Dans le script de gauche, on voit qu'un bloc est répété 2 fois. Dans ce cas-là, une boucle `for` permet de gérer cette répétition.<br>La variable `i` prendra les valeurs 0 puis 1 dans notre cas, la boucle sera effectuée 2 fois. |

**Pourquoi c'est cool ?** Dans la version sans boucle, on répète un bloc 2 fois. Avec `for i in range(2)`, Python s'en charge automatiquement. Résultat : code plus court, moins d'erreurs, et plus facile à modifier (change juste le `range` pour plus de répétitions !).

**Astuce de pro** 💡 **:** L'indentation (4 espaces ou une tab) est ton amie ! Elle dit à Python : "Hey, ce code est DANS la boucle !" Sois rigoureux, sinon... bug party ! 🐞

**Exercice fun :** Modifie tes scripts du "carré sans coin" et de la "spirale" pour utiliser une boucle `for`. Ajoute un commentaire expliquant pourquoi tu l'as fait. Bonus : Fais tourner la spirale plus de fois et vois ce qui se passe ! Partage ton dessin le plus fou en classe.

## 2. Utilisation de input() pour Personnaliser les Dimensions 🎨

Et si ton dessin devenait interactif ? Genre, "Hey utilisateur, choisis la taille et la couleur !" C'est là que `input()` entre en scène – comme un menu dans un jeu vidéo.

**Exemple à tester (copie, colle, et joue !) :**

```
from turtle import *

# Demande à l'utilisateur - fun et personnalisé !
longueur = int(input("Entrez la longueur du rectangle (ex: 100) : "))
largeur = int(input("Entrez la largeur du rectangle (ex: 50) : "))
couleur = input("De quelle couleur veux-tu ton rectangle ? (ex: red, blue, green) : ")

color(couleur)  # Applique la couleur choisie

for i in range(2):
    forward(longueur)  # Utilise la variable !
    left(90)
    forward(largeur)
    left(90)

hideturtle()  # Bye bye tortue
done()        # Admire ton œuvre !
```

**Analyse rapide (comme un détective 🕵️) :**

- `input("Message")` pose une question à l'utilisateur et stocke la réponse (comme une chaîne de caractères, aka "str").
- `int()` convertit ça en nombre entier (sinon, boom – erreur ! Voir TP sur les types de données).
- Résultat : Ton rectangle est unique à chaque run. Essaie des couleurs folles comme "purple" ou "hotpink" !

**Remarque importante !** ⚠️ Sans `int()`, la réponse est du texte – pas utilisable pour des maths. Toujours convertir pour les nombres !

**Exercice fun :** Ajoute des `input()` dans tes scripts du carré sans coin et de la spirale. Laisse l'utilisateur choisir la taille, la couleur, et peut-être le nombre de tours pour la spirale.
Bonus : Ajoute un `input()` pour l'épaisseur du trait (`pensize()`). Qui fera la spirale la plus épique ? 🏆

## 3. Création de Fonctions avec def 🛠️

Les fonctions, c'est comme des recettes réutilisables : écris une fois, utilise partout ! Parfait pour ne pas réécrire le même code.

**Exemple pour le rectangle (crée un fichier `fonction_rectangle.py` et teste) :**

```
from turtle import *

def rectangle(longueur, largeur, couleur="black"):  # Paramètres avec valeur par défaut pour la couleur
    color(couleur)  # Applique la couleur
    for i in range(2):
        forward(longueur)  # Utilise les variables !
        left(90)
        forward(largeur)
        left(90)

# Teste la fonction ici ou en console !
rectangle(150, 70, "blue")  # Appelle-la avec tes valeurs
hideturtle()
done()
```

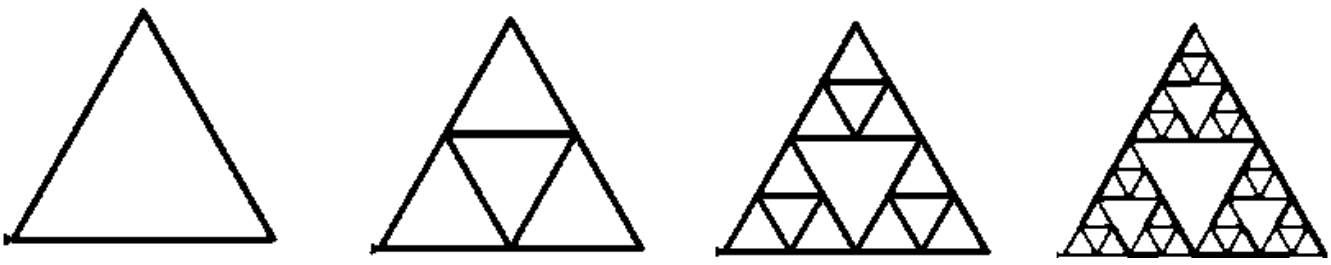
**Comment ça marche ?**

- `def nom_fonction(params)`: définit ta fonction (les paramètres ne sont pas obligatoires).
- Indente le code dedans (encore l'indentation !).
- Appelle-la avec `rectangle(150, 70, "blue")` – et pouf, dessin ! Ajoute plus de params comme `pensize` pour l'épaisseur.

**Astuce fun** 💥 **:** En console, tape `rectangle(200, 100, "red")` pour un nouveau dessin sans relancer tout. C'est magique !

**Exercice fun :**
Crée deux fichiers : `fonction_carre_sans_coin.py` et `fonction_spirale.py`. Implémente des fonctions avec params (taille, couleur, épaisseur...).
**Bonus :** Ajoute un param pour le nombre de côtés ou de tours. Teste avec des valeurs extrêmes – qui crash la tortue en premier ? 😜

**Défi Ultime : La Fractale Mystère** 🌟
Observe la construction de cette fractale (imagine une belle image ici – probablement une courbe de Koch ou un arbre fractal, avec des phases qui se complexifient !).



**Ton défi de ninja** 🥷 **:** Écris un programme qui dessine cette fractale à la phase 4... et au-delà ! Utilise la récursion (fonctions qui s'appellent elles-mêmes), des boucles, et des input() pour choisir la phase.